

StaticGreedy: solving the apparent scalability-accuracy dilemma in influence maximization

Suqi Cheng Huawei Shen Junming Huang Guoqing Zhang Xueqi Cheng
 Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
 {chengsuqi, shenhuawei, huangjunming, gqzhang, cxq}@ict.ac.cn

ABSTRACT

Influence maximization, defined as the problem of finding a set of seed nodes maximizing the spread of influence, is crucial to viral marketing on social networks. For practical viral marketing on large-scale social networks, it is required that influence maximization algorithms have both guaranteed accuracy and high scalability. However, existing algorithms suffer an apparent scalability-accuracy dilemma: Greedy algorithm and its improvements have guaranteed accuracy but are not scalable, while the accuracy of scalable heuristic algorithms is unstable and not guaranteed.

In this paper, we focus on resolving this scalability-accuracy dilemma. We first find that the submodularity is unguaranteed in existing implementations of greedy algorithm, caused by the independence among Monte Carlo simulations conducted in different iterations of greedy algorithm. A large number of Monte Carlo simulations are thus required in existing greedy algorithms to alleviate the impact of unguaranteed submodularity. Motivated by this critical finding, we propose a static greedy algorithm to strictly guarantee the submodularity property, by reusing the results of Monte Carlo simulations during the whole process of greedy algorithm. As a result, the proposed algorithm achieves the same accuracy with the state-of-the-art greedy algorithms, while the number of Monte Carlo simulations needed is dramatically reduced by two orders of magnitude. Moreover, we give a dynamic update strategy to further improve the static greedy algorithm, by applying which our algorithm becomes comparable to the most scalable heuristic algorithm.

Categories and Subject Descriptors

F.2.2 [Analysis of Algorithms and Problem Complexity]: Non-numerical Algorithms and Problems; D.2.8 [Software Engineering]: Metrics—*complexity measures, performance measures*

General Terms

Algorithms, Experiments, Performance

Keywords

influence maximization, independent cascade model, greedy algorithm, scalability, social networks

1. INTRODUCTION

We are witnessing the increasing prosperity of online social network sites and social media sites, where people are connected by heterogeneous social relationships. These online social networks provide convenient platforms for information dissemination and marketing, allowing ideas and behaviors to flow along the social relationships in the effective word-of-mouth manner. Many companies have made efforts to popularize or promote their brands or products on online social networks by launching campaigns akin to viral marketing. The success of viral marketing is rooted in the interpersonal influence, which has been empirically studied in various contexts [1, 2, 3, 4, 5].

Influence maximization, formulated as a discrete optimization problem by Kempe, Kleinberg and Tardos [6], is a fundamental problem for viral marketing. It aims to find a fixed-sized set of seed nodes, which maximizes the spread of influence on a social network. Its evaluation is the expected number of nodes influenced by the seed set, which is referred to as *influence spread*. The solution of influence maximization problem is closely related to influence spread models, which are used to model the process of influence spread. Two commonly-used models are the independent cascade model and the linear threshold model. With respect to the two models, Kempe et al. [6] prove that the influence maximization problem is NP-hard, and they present a greedy approximation algorithm guaranteeing that the obtained influence spread is no less than $1 - 1/e - \epsilon$ of the optimal value, where ϵ depends on the accuracy of influence spread estimation. Such approximation guarantee is further extended to general threshold model [7]. Since there is no efficient algorithm to obtain the exact influence spread of a given seed set [8] [9], the average influence spread over a sufficiently large number of Monte Carlo simulations is widely used to approximate the exact value.

However, the general greedy algorithm proposed by kempe et al. is not scalable for involving too many Monte Carlo simulations, and this limits its application to networks with small or moderate size. To overcome this problem, many

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Copyright 20XX ACM X-XXXXX-XX-X/XX/XX ...\$15.00.

efforts have been made to improve the scalability of this general greedy algorithm [10, 11, 12, 13, 14, 15, 16]. These improvements are obtained along two directions. For the first one, some researchers attempt to reduce the times of influence spread estimations, i.e., computing the influence spread of certain node sets. For example, a “lazy-forward” strategy [10] is proposed to effectively reduce the number of candidate nodes. Yet such reduction quickly hits its limit, caused by the requirements of guaranteed accuracy, which suggests a large number of Monte Carlo simulations in a single influence spread estimation. For the other direction, various heuristics are proposed to use more efficient methods for influence spread estimation, rather than Monte Carlo simulations. In the representative work conducted by Chen et al. [8], the maximum influence paths between every pair of nodes are taken as delegates to estimate the influence spread for each node. However, the increase of scalability is obtained at the cost of unstable or unguaranteed accuracy. In a word, existing algorithms for influence maximization are facing the scalability-accuracy dilemma.

This paper focuses on resolving the scalability-accuracy dilemma of influence maximization with respect to the independent cascade model. We analyze the essential cause of the scalability-accuracy dilemma, and then propose a static greedy algorithm to combat it. Moreover, we further improve the scalability of the static greedy algorithm by a dynamic update strategy. The contributions of this paper are summarized as follows:

- We point out that the submodularity property is not guaranteed in existing implementations of greedy algorithm, which directly leads to the inefficiency of previous greedy algorithms. This critical finding renews our knowledge about the greedy algorithm for influence maximization and opens the door to resolve the accuracy-scalability dilemma.
- We propose a static greedy algorithm to strictly guarantee the submodularity property by reusing the results of Monte Carlo simulations during the whole process of greedy algorithm. The proposed algorithm achieves the guaranteed accuracy with the number of Monte Carlo simulations dramatically reduced, thus it effectively improves the scalability of the greedy algorithm.
- We give a strategy to further speedup our static greedy algorithm by dynamically updating the marginal gain of the candidate nodes. This strategy, which takes the advantage of static results of Monte Carlo simulations, improves the efficiency of our static greedy algorithm in a way that enables it to run 2-7 times faster than the CELF optimized static greedy algorithm in the testing of social networks used in this paper, and makes it comparable to the most scalable heuristic algorithm.

2. RELATED WORK

Influence maximization was first studied by Domingos and Richardson from the algorithmic perspective [1, 2], and was then formulated as a discrete optimization problem by Kempe et al. [6]. They also propose a greedy algorithm, with guaranteed accuracy caused by the monotone and submodularity

properties of the objective function of influence maximization problem. However, this greedy algorithm is inefficient and not scalable to large scale social networks.

Thus, several studies devote to optimize Kempe’s greedy algorithm without affecting guaranteed accuracy. Leskovec et al. [10] propose the “cost-effective lazy forward” strategy, namely CELF, for selecting new seeds by further exploiting the submodularity property of influence maximization. The CELF strategy can greatly reduce the number of evaluations on the influence spread of nodes. This strategy is further improved to a CELF++ strategy [16], which suggests simultaneously calculating the influence spread used in successive iterations of greedy algorithm. NewGreedy algorithm [11] reuses the results of Monte Carlo simulations to estimate the influence spread for all candidate nodes in the same iteration. It has been further developed into Mixed-Greedy algorithm to integrate the advantages of the CELF strategy and the NewGreedy algorithm.

Unfortunately, those improved greedy algorithms are still inefficient for involving too many Monte Carlo simulations for influence spread estimation. Hence, several heuristic algorithms for the independent cascade model are proposed to improve the scalability of greedy algorithm by simplifying influence spread estimation. Chen et al. [11] suggest a degree discount heuristics for influence maximization on uniform independent cascade model. Wang et al. [13] divide a network into communities and conduct Monte Carlo simulations within each community instead of the whole network. Luo et al. [17] propose to conduct the greedy algorithm on a small set of nodes, which consists of the top nodes ranked by PageRank algorithm on social network. Kimura and Saito [12] propose the shortest-path based influence cascade models and provide efficient algorithms to compute the influence spread under these models. Instead of using the simple shortest path, PMIA algorithm [8] [18] uses maximum influence paths for influence spread estimation, and this algorithm is believed to be the best heuristic algorithm so far. However, these heuristics may violate the guaranteed accuracy of greedy algorithm and thus one may concern about the reliability of these heuristics.

In addition, several influence maximization algorithms are beyond the framework of greedy algorithm. Jiang et al. [15] suggest a simulated annealing approach with several heuristics to speed up the computation of the influence spread. Narayanan and Narahari [14] give a way to improve the scalability of influence maximization using the concept of Shapley value borrowed from the cooperative game theory. Mathioudakis et al. [19] suggest removing some unimportant edges for influence propagation to accelerate influence computation algorithms.

3. STATIC GREEDY ALGORITHM

3.1 Influence maximization problem

We consider the influence maximization problem with respect to the independent cascade(IC) model. For a directed graph $G = (V, E)$, each edge $\langle u, v \rangle \in E$ is associated with a propagation probability $p(u, v)$, denoting the probability that v is activated by u through the edge $\langle u, v \rangle$ after u is activated. In the IC model, whether u activates v is fully determined by $p(u, v)$, which is independent from the probabilities

associated with other edges. A node cannot become inactive once it becomes active. Each node has only one chance to activate each of its inactive neighbors. Given a seed set S , its influence spread $I(S)$ is defined as the expected number of nodes eventually activated. Influence maximization problem aims at finding the set S which maximizes $I(S)$, under the constraint that the size of S is no larger than a predefined positive integer k .

To resolve the influence maximization problem, a method is needed to evaluate $I(S)$ for a given S . However, it is intractable to exactly compute $I(S)$ on a typically sized graph. In practice, Monte Carlo simulation is employed to estimate $I(S)$, and it can be implemented in two different ways as follows:

- **Simulation.** The influence spread is obtained by directly simulating the random process of influence cascade from a seed set. Given a set S , the simulation is conducted as follows. Let A_i denote the set of nodes activated in the i -th iteration and thus $A_0 = S$. Then, at the $(i + 1)$ -th iteration, for each node $u \in A_i$, it attempts to activate its inactive neighbors and successfully activates v with the probability $p(u, v)$ associated with the edge $\langle u, v \rangle$. This process is repeated until no nodes are newly activated. For each random cascade, the number of eventually activated nodes is the influence spread of this single simulation. Finally, by running such random process of influence cascade for many times, we can estimate the influence spread $I(S)$ by averaging over all these simulations.
- **Snapshot.** According to the characteristic of the IC model, *snapshots* can be obtained for the influence propagation graph G a priori. A snapshot is a graph G' , where an edge $\langle u, v \rangle$ is remained with the probability $p(u, v)$. Note that each snapshot is an instance of the probability space comprising the graphs obtained by sampling the influence propagation graph G . For each snapshot G' , the influence spread of S is the number of nodes reachable from S . Then, $I(S)$ can be obtained by averaging over many snapshots.

The results of above two methods are essentially equivalent and each has its own unique advantage. For estimating the influence spread of a given node set, the simulation method is faster, because it only needs to explore a small portion of edges, while the snapshot method has to check all the edges in the graph. If we need to estimate the influence spread of many sets, the snapshot method outperforms the cascade method in terms of time complexity, since we can reuse the snapshots to calculate the influence spread for all candidates.

3.2 Submodularity in greedy algorithms

For greedy algorithms of influence maximization, to guarantee the approximation within a factor of $1 - 1/e$, it is required that the influence spread function $I(\cdot)$ is a monotone and submodular function, and its value can be evaluated exactly [20]. We say that a function $f(\cdot)$ is monotone if $f(S \cup \{v\}) \geq f(S)$ for any set S and any element $v \notin S$, and $f(\cdot)$ is submodular iff $f(S \cup \{v\}) - f(S) \geq f(T \cup \{v\}) - f(T)$ when $S \subseteq T$. The submodularity property is also explained as a natural “diminishing return” property. Although $I(\cdot)$

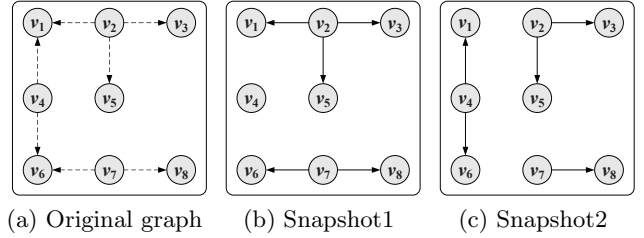


Figure 1: Illustrations of unguaranteed submodularity property.

is proven to be monotone and submodular [6], the value of $I(\cdot)$ cannot be calculated exactly. That is just the difficulty faced by implementing greedy algorithm for the influence maximization. In practice, Monte Carlo simulations is employed to approximately estimate $I(\cdot)$.

Unfortunately, things become different when Monte Carlo simulation is employed. In existing implementations of greedy algorithms, Monte Carlo simulations are conducted independently among different iterations. The spread along an edge (u, v) may fail in one Monte Carlo simulation and succeed in the other Monte Carlo simulation. As a result, the marginal gain $I(S_i \cup \{v\}) - I(S_i)$ from adding v to the seed set S_i in the i -th iteration might be lower than the marginal gain $I(S_{i+1} \cup \{v\}) - I(S_{i+1})$ from adding v in the $(i + 1)$ -th iteration. Therefore, after using Monte Carlo simulation, the submodularity property of the influence spread function $I(\cdot)$ becomes unguaranteed. Moreover, the monotone property also suffers the same problem.

Figure 1 gives a simple example to illustrate the unguaranteed submodularity caused by Monte Carlo simulation. Figure 1(a) depicts a graph with each edge associated with a certain propagation probability (e.g., 0.5). Figure 1(b) represents the result of one Monte Carlo simulation by remaining only the edges along which the spread succeeds. Figure 1(c) represents the result of another Monte Carlo simulation. Suppose Figure 1(b) is used in the first iteration in a greedy algorithm and Figure 1(c) is used in the second iteration. Starting from an empty seed set $S_0 = \phi$, the greedy algorithm selects node v_2 as the seed node in the first iteration since v_2 has the largest influence spread. In second iteration, we use the result of Monte Carlo simulation in Figure 1(c) and the marginal gain of adding v_4 is $I(S_2 \cup \{v_4\}) - I(S_2) = I(\{v_2, v_4\}) - I(\{v_2\}) = 2$. However, the marginal gain of v_4 is $I(S_1 \cup \{v_4\}) - I(S_1) = I(\{v_2, v_4\}) - I(\{v_2\}) = 0$ according to the result of Monte Carlo simulation in Figure 1(b). That is to say, the marginal gain of v_4 in the first iteration is smaller than in the second iteration. This is contradictory to the submodularity property.

To avoid the unguaranteed submodularity caused by the independence between Monte Carlo simulations conducted among different iterations, existing greedy algorithms have to use an extremely large number R (typically $R = 10,000$ or $R = 20,000$) of Monte Carlo simulations in each iteration. These algorithms want to guarantee the submodularity and monotone properties by closely approximate the exact influence spread. However, the submodularity and monotone properties can only be guaranteed with a certain probability in this way because of the finite number

of Monte Carlo simulations. This poses concern about the accuracy of existing greedy algorithms with unguaranteed submodularity and monotone properties. In addition, to improve the accuracy, existing greedy algorithms have to use a huge number of Monte Carlo simulations. This then results in expensive computational cost and limits the application of these greedy algorithms on real world social networks with millions of nodes. Therefore, these algorithms suffer a scalability-accuracy dilemma and this dilemma has roots in the independence between Monte Carlo simulations conducted in different iterations of these greedy algorithms.

3.3 Description of static greedy algorithm

We have pointed out that, the key for combating the scalability-accuracy dilemma is the independence between Monte Carlo simulations conducted in different iterations of greedy algorithms. Along this line, we propose a new greedy algorithm which shares the results of Monte Carlo simulations in all the iterations of greedy algorithm. We call our algorithm a static greedy algorithm, namely **StaticGreedy**, because the results of Monte Carlo simulations can be generated a priori in terms of the snapshot manner, as described in Section 3.1 and are kept static in the whole greedy algorithm.

Given an underlying social network G and a positive integer k , the StaticGreedy algorithm seeks for a seed set S to maximize the influence spread $I(S)$ according to the following process:

1. Static snapshots: Randomly sampling R snapshots from the underlying social network G . In each snapshot, each edge $\langle u, v \rangle$ is sampled with its associated probability $p(u, v)$;
2. Greedy selection: Start from an empty seed set S , then iteratively add one node a time into S such that the node provides the largest marginal gain of $I(S)$, which is estimated on the R snapshots. The process continues until k nodes have been selected.

The StaticGreedy algorithm is formally described in Algorithm 1. Two main differences between this algorithm and existing greedy algorithms include: (1) Monte Carlo simulations are conducted in static snapshot manner, which are sampled before the greedy process of selecting seed nodes, as is shown in line 2 to 4; (2) The same set of snapshots are reused in every iteration to estimate the influence spread $I(S)$, where explains the meaning of “static”.

The StaticGreedy algorithm has two benefits: 1) the accuracy is guaranteed since the submodularity and monotone properties of influence spread function are strictly guaranteed, 2) the StaticGreedy algorithm is highly scalable. In StaticGreedy, it is not required that the influence spread should be accurately approximated by a large number R of Monte Carlo simulations. Thus, R could be significantly reduced to a low magnitude. Now the requirements for R is that R snapshots can provide a sufficient and representative delegate of the underlying social network. Roughly speaking, this only requires that each edge can be observed in the R snapshots at least once. Thus, R is in the magnitude of $O(1/p_{\min})$, where p_{\min} is the smallest propagation probability on all edges. For a typical social network with $p = 0.01$

Algorithm 1 StaticGreedy(G, k, R)

```

1: initialize  $S = \emptyset$ 
2: for  $i = 1$  to  $R$  do
3:   generate  $G'_i$  by removing each edge  $\langle u, v \rangle$  from  $G$  with
     probability  $1 - p(u, v)$ 
4: end for
5: for  $i = 1$  to  $k$  do
6:   set  $s_v = 0$  for all  $v \in V \setminus S$ 
7:   for  $j = 1$  to  $R$  do
8:     for all  $v \in V \setminus S$  do
9:        $s_v += |R(G'_j, S \cup \{v\})|$ 
10:    end for
11:  end for
12:   $S = S \cup \{\arg \max_{v \in V \setminus S} \{s_v / R\}\}$ 
13: end for
14: output  $S$ 

```

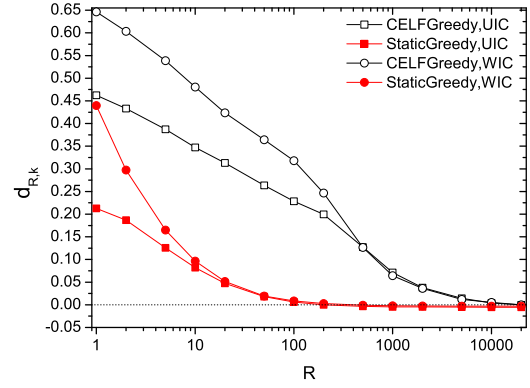


Figure 2: The relationship between $d_{R,k}$ and R of CELFGreedy and StaticGreedy on NetHEPT network.

or so, hundreds of snapshots could be enough to represent the social network. This value is much less than what the existing algorithms require, i.e., typically in the magnitude of 10,000.

3.4 Analysis of the StaticGreedy algorithm

3.4.1 Accuracy

To clarify the performance of the StaticGreedy algorithm compared with the original greedy algorithm, we illustrate the accuracy of these algorithms with respect to the number R of Monte Carlo simulations on a benchmark network NetHEPT. This network consists of tens of thousands of physics researchers and their co-authorship relations. The employed baseline greedy algorithm is the CELFGreedy, which is the general greedy algorithm with CELF optimization. In addition, we choose two commonly-used IC models: the uniform independent cascade (UIC) model with $p = 0.01$ and the weighted independent cascade (WIC) model introduced in Ref. [6] with a setting that $p(u, v) = 1/k_v$, where k_v is the indegree of node v .

Since the optimal influence spread is unknown to us, the

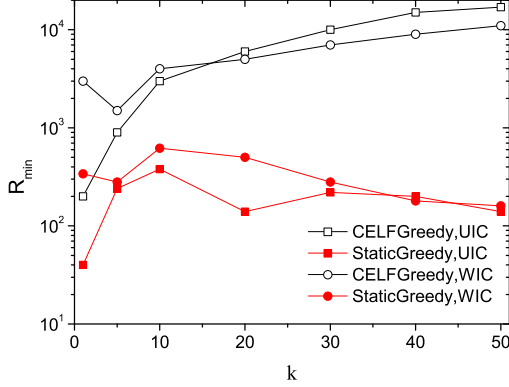


Figure 3: Minimal number of snapshots needed to accurately find a solution.

ground truth we use here for each value of set size k is the influence spread of the solution S_k^* obtained by CELFGreedy algorithm with typical setting, i.e., $R = 20,000$. To evaluate the relative difference between the influence spread obtained by a greedy algorithm and the ground truth, we use a measure $d_{R,k}$ defined as

$$d_{R,k} = \frac{I(S_k^*) - I(S_{R,k})}{I(S_k^*)},$$

where $S_{R,k}$ is the set of seed nodes obtained by a greedy algorithm with a given R , and k indicates the size of the seed set. For a given R , we run both the StaticGreedy algorithm and the CELFGreedy algorithm for 50 times to calculate the average relative difference. We only report the results with $k = 50$ since the results for other k are similar.

As shown in Figure 2, with R increases, the StaticGreedy algorithm quickly approaches the ground truth while the CELFGreedy algorithm converges slowly. For the same R , the accuracy of the StaticGreedy algorithm consistently outperforms the CELFGreedy algorithm. This confirms that the StaticGreedy algorithm can achieve good accuracy even when the number R of Monte Carlo simulations is small, e.g., $R = 100$.

We further evaluate the accuracy of the StaticGreedy algorithm with respect to the size k of seed set. For this purpose, we define R_{min} as the minimal R satisfying $d_{R,k} \leq 0.005$ and depicts R_{min} with respect to k . As shown in Figure 3, for all the tested values of k , the StaticGreedy algorithm always has a small R_{min} , greatly smaller than the value of R_{min} for the CELFGreedy algorithm.

3.4.2 Scalability

Now we analyze the time complexity of the StaticGreedy algorithm. For clarity, we use R to denote the number of Monte Carlo simulations required by existing greedy algorithms and use R' to denote the number of Monte Carlo simulations required by our StaticGreedy algorithm. In addition, n is the number of nodes in the underlying influence network, m is the number of edges in the network, m' is the average number of active edges in the snapshots obtained by sampling the influence network, and k is the number

Table 1: Time and space complexity of algorithms

Algorithms	Time complexity	Space complexity
GeneralGreedy	$O(knRm)$	$O(m)$
StaticGreedy	$O(R'm + knR'm')$	$O(R'm')$

of seed nodes. For the StaticGreedy algorithm, the time complexity includes two parts: firstly, the time complexity of generating R' snapshots is $O(R'm)$; secondly, it takes $O(knR'm')$ time to select seed nodes in greedy manner on those static snapshots. Thus, the total time complexity is $O(R'm + knR'm')$. For the space complexity of StaticGreedy algorithm is $O(R'm')$, which is used to store the R' snapshots. The comparison with the general greedy algorithm [6] is given in Table 1.

The time complexity of the StaticGreedy algorithm can be further reduced by employing the CELF optimization and other optimization strategies. In the next section, we give a dynamic update strategy to improve the efficiency of the StaticGreedy algorithm.

3.4.3 Discussions

In Ref. [11], the authors propose to reuse snapshots within the same iteration. Their motivation is to reduce the computational cost through simultaneously estimating the influence spread of many seed sets. However, reusing snapshots within the same iteration cannot guarantee the submodularity and monotone properties as done by the StaticGreedy algorithm. Thus a large number of snapshots are required for each iteration of the greedy algorithm.

In addition, how do we determine the minimum R for a specific network and a given spread model? What are the factors affecting R in StaticGreedy or previous greedy algorithms? We leave these interesting questions as open problems in the future.

4. SPEEDING UP THE STATICGREEDY

In this section, we propose a dynamic update strategy to further speed up the static greedy algorithm. This strategy exploits the advantage of static snapshots and calculates the marginal gain in an efficient incremental manner. Specifically, when a node v^* is selected as a seed node, we directly discount the marginal gain of other nodes by the marginal gain shared by these nodes and v^* .

For a snapshot G'_i , we use $R(G'_i, v)$ to denote the set of nodes which can be reachable from v and use $U(G'_i, v)$ to denote the set of nodes from which v can be reached. In the first iteration, the marginal gain of v is $|R(G'_i, v)|$. In our dynamic update strategy, when v^* is selected as a seed node, we find the set $U(G'_i, w)$ for each node $w \in R(G'_i, v^*)$. Then, for every $u \in U(G'_i, w)$, we delete w from $R(G'_i, u)$. Now, the size of $R(G'_i, u)$ reflects the marginal gain of u in the next iteration. In this way, we can maintain a dynamically updated marginal gain for each node in order to avoid calculating the marginal gain from scratch. The detailed implementation of the static algorithm, called as **StaticGreedyDU**, is given in Algorithm 2.

Now we analyze the time and space complexity of the StaticGreedyDU algorithm. For undirected graphs, $R(G'_i, v)$ is

the same to $U(G'_i, v)$. We only need to store the information of connected components for each snapshot. Thus, the space complexity is $O(R'n)$. It takes $O(R'm)$ time to generate R' snapshots and calculate the initial marginal gain, and $O(kn)$ time to update information for all the related nodes. Then, the total time complexity is $O(R'm + kn)$. For directed graphs, let $n_T = \max_{v \in V} R(G'_i, v)$, $n_U = \max_{v \in V} U(G'_i, v)$. Since it needs to store $R(G'_i, v)$ and $U(G'_i, v)$ for each node, the space complexity is $O(R'nn_T + R'nn_U)$. Assume the maximum running time to compute $R(G'_i, v)$ and $U(G'_i, v)$ is t_T and t_U respectively. It takes $O(R'm)$ time to generate snapshots, $O(R'nt_T + R'nt_U)$ time to compute the initial incremental influence spread, and $O(kR'n_Tn_U)$ time to update information. Hence, the total time complexity is $O(R'm + R'nt_T + R'nt_U + kR'n_Tn_U)$. Note that n_T , n_U , t_T and t_U are usually very small in real world networks which are sparse.

Algorithm 2 StaticGreedyDU(G, k, R)

```

1: initialize  $S = \emptyset$ 
2: set the marginal gain  $s_v = 0$  for all  $v \in V$ 
3: for  $i = 1$  to  $R$  do
4:   generate  $G'_i$ 
5:   compute and record  $R(G'_i, v)$  and  $U(G'_i, v)$  for all  $v \in V$ 
6:   for each node  $v \in V$  do
7:      $s_v += |R(G'_i, v)|$ 
8:   end for
9: end for
10: for  $r = 1$  to  $k$  do
11:    $v^* = \arg \max_{v \in V \setminus S} \{s_v\}$ 
12:    $S = S \cup \{v^*\}$ 
13:   for  $i = 1$  to  $R$  do
14:     for each node  $w \in R(G'_i, v^*)$  do
15:       for each node  $u \in U(G'_i, w)$  do
16:         delete  $w$  from  $R(G'_i, u)$ 
17:          $s_u = s_u - 1$ 
18:       end for
19:     end for
20:   end for
21: end for
22: output  $S$ .
```

5. EXPERIMENT

In this section, we conduct experiments on several real-world networks to compare our StaticGreedy algorithm with a number of existing algorithms. The experiments aim at illustrating the performance of our algorithm comparing to other algorithms from the following two aspects: (a) accuracy at finding the seed nodes maximizing the influence spread, (b) scalability.

5.1 Experiment setup

Datasets. Six real world networks are employed to demonstrate the performance of our algorithms by comparing with other existing algorithms. These networks include three scientific collaboration networks and three online social networks. For the three scientific collaboration networks, namely

Table 2: Statistics of six test real world networks.

Datasets	#Nodes	#Edges	Directed?
NetHEPT	15K	59K	undirected
NetPHY	37K	231K	undirected
DBLP	655K	2M	undirected
Epinions	76K	509K	directed
Slashdot	77K	905K	directed
Douban	552K	22M	directed

NetHEPT, NetPHY, and DBLP¹, nodes are authors and edges represent the coauthor relationships among authors. All of them are undirected networks. NetHEPT is extracted from “High Energy Physics - Theory” section of the e-print arXiv (<http://www.arXiv.org>) between 1991 and 2003. NetPHY is constructed from the full paper list of the “Physics” section. DBLP, much larger than the former two scientific collaboration networks, is extracted from the DBLP Computer Science Bibliography Database maintained by Michael Ley. The three online social networks, namely Epinions, Slashdot, and Douban², are collected from the websites Epinions.com, Slashdot.com, and Douban.com. In Epinions, an edge (u, v) in this network means u trust v . Slashdot is a friend network extracted from a technology-related news website Slashdot.com. The last dataset Douban [5] is collected from douban.com, where users can rate books and movies, and follow other users. The edges in this network represent followships among users. All the three online social networks are directed networks. We choose the above six networks since they cover a variety of networks with sizes ranging from tens of thousands of edges to millions of edges. Some basic statistics about these networks are given in Table 2.

Cascade Models. The adopted cascade models are still the two commonly-used IC model: the UIC model and WIC model, which we have used in section 3.4.1. For the UIC model, we set the propagation probability $p = 0.001$ for Douban and $p = 0.01$ for other networks. That is because the average degree of Douban is nearly ten times than that of others. For the WIC model, $p(u, v) = 1/k_v$, where k_v is the indegree of node v .

Algorithms. We compare our StaticGreedy algorithm and its improved versions with both CELFGreedy and several heuristic algorithms, including PMIA, DegreeDiscount and Degree. For StaticGreedy, we set $R = 100$ as default, in other word, 100 snapshots are employed for all the datasets. Thus, StaticGreedy algorithm may have better accuracy if we improve the number of employed snapshots. For CELFGreedy, R is set to 20,000. The PMIA algorithm has a tunable parameter θ . We set the value of θ as done in Ref. [8]. The DegreeDiscount heuristic [11] is developed for the UIC model with the propagation probability $p = 0.001$

¹The three networks are downloaded from <http://research.microsoft.com/en-us/people/weic/>. These scientific collaboration networks are actually multigraphs, with parallel edges between two nodes denoting the number of papers coauthored by the two authors.

²The former two networks can be downloaded from <http://snap.stanford.edu/data/>. The last one can be obtained on requirement via email to the authors.

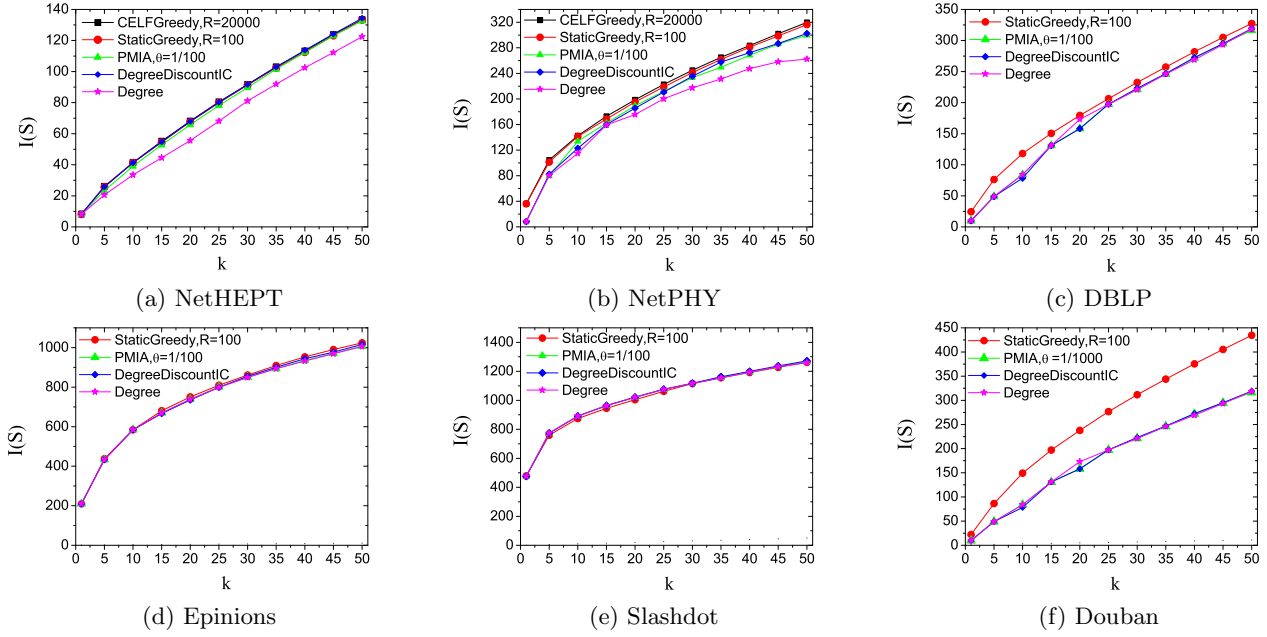


Figure 4: Influence spread under UIC model on six datasets.

for Douban and $p = 0.01$ for other cases. For Degree heuristic, it simply selects seed nodes according to the degree of nodes.

Since the PMIA algorithm is the state-of-the-art heuristic [8], we do not consider other heuristics, including distance centrality, betweenness centrality, and PageRank heuristic.

5.2 Experimental results

We run tests on the six datasets and two IC models, i.e., UIC model and WIC model. The tested seed size k are 1, 5, 10, 15, 20, 25, 30, 35, 40, 45, and 50. For the comparison of running time, we only consider the seed size $k = 50$. In addition, the experiments are conducted on a server with 2.0GHz Quad-Core Intel Xeon X7550 and 64G memory.

5.2.1 Accuracy comparison

We first test the accuracy of StaticGreedy algorithms by showing the influence spread of the obtained set of nodes. For every obtained seed set, 20,000 Monte Carlo simulations are used to evaluate its influence spread. Figure 4 shows the experimental results on influence spread for the six datasets on the UIC model. As shown in Figure 4(a) and Figure 4(b), the CELFGreedy algorithm provides the best influence spread on the moderate sized networks NetHEPT and NetPHY where the CELFGreedy algorithm is still feasible to run. On the dataset NetHEPT, all the algorithms except the Degree heuristic algorithm have the influence spread similar to the CELFGreedy algorithm. However, on the dataset NetPHY, the differences among these algorithms become visible. StaticGreedy algorithm is still very close to the CELFGreedy algorithm and outperforms all the other algorithms. In fact, the difference between StaticGreedy algorithm and the CELFGreedy algorithm is less than 2%. For the rest networks with large scale where the CELFGreedy algorithm is infeasible due to its high computation cost, we compare StaticGreedy algorithm with the other three base-

line algorithms. We can see that StaticGreedy algorithm always has the best accuracy compared with other algorithms. In particular, for the DBLP and Douban datasets, StaticGreedy algorithm significantly outperforms the competing algorithms.

We further test StaticGreedy algorithm on the six test datasets with respect to the WIC model. For the moderate sized networks NetHEPT and NetPHY where CELFGreedy is still feasible to run, as shown in Figure 5(a) and Figure 5(b), StaticGreedy algorithm has almost the same influence spread to the CELFGreedy algorithm, which is the most accurate greedy algorithm. Moreover, StaticGreedy algorithm outperforms the other algorithms with a visible gap. For the DBLP, Epinions, Slashdot and Douban networks with large scale, StaticGreedy algorithm has consistent accuracy with the other three baseline algorithms while the CELFGreedy algorithm is not scalable to these networks.

As demonstrated by the results on the six test networks with both the UIC and WIC models, StaticGreedy algorithm has guaranteed accuracy as the original greedy algorithm and outperforms the state-of-the-art heuristic algorithms. More important, compared with the original greedy algorithm, the guaranteed accuracy of our static greedy algorithm is obtained with the number of Monte Carlo simulations dramatically reduced by two orders of magnitude.

5.2.2 Running time comparison

We now test the running time of StaticGreedy algorithm and the competing algorithms. With respect to StaticGreedy, we test the running time of both the StaticGreedyDU algorithm and the StaticGreedy algorithm with CELF optimization, denoted as StaticGreedyCELF. Figure 6 shows the experimental results. For the six networks, StaticGreedyDU always 2-7 times faster than StaticGreedyCELF, and such seep difference is even more significant for DBLP. For the moderate sized datasets, i.e., NetHEPT and NetPHY, the

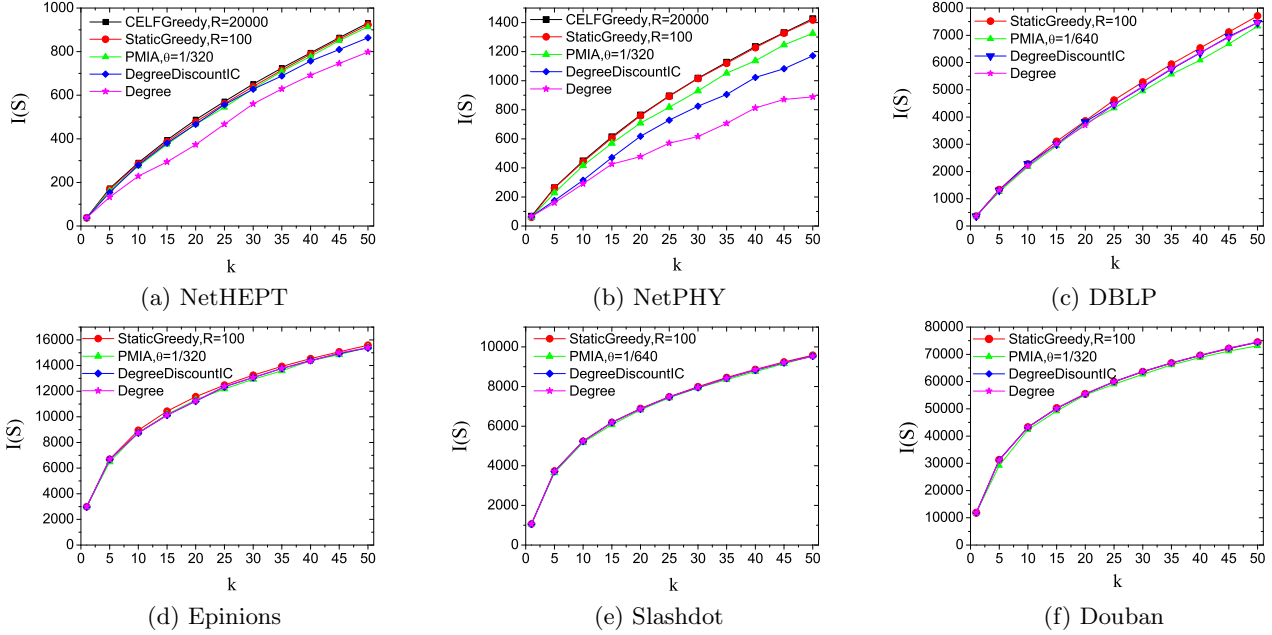


Figure 5: Influence spread under WIC model on six datasets.

CELFGreedy algorithm is already quite slow. The CELFGreedy algorithm requires several hours while our static greedy algorithms only take several seconds. StaticGreedyDU and StaticGreedyCELF both reduce the running time by three orders of magnitude, compared with the CELFGreedy algorithm. More importantly, the reduction of running time is obtained by our static greedy algorithm without affecting the guaranteed accuracy. The time cost of our two static greedy algorithms is comparable to the PMIA algorithm, which is the most scalable heuristic algorithm. Note that the accuracy of the PMIA algorithm is unguaranteed in theory. Moreover, StaticGreedyDU algorithm even outperforms the PMIA algorithm on three large scale networks, Epinions, Slashdot and Douban. It seems that our algorithm has the potential advantage on large scale networks compared with the PMIA algorithm. For the Degree and DegreeDiscount algorithms, they have higher scalability than our method. However, this benefit is obtained at the cost of the decrease of the obtained influence spread.

6. CONCLUSION AND FUTURE WORK

In this paper, we have analyzed the scalability-accuracy dilemma of the greedy algorithm, which has roots in the unguaranteed submodularity property in existing implementations. To combat this problem, we propose a static greedy algorithm to remove the independence among Monte Carlo simulations in different iterations. The submodularity property of influence spread is strictly guaranteed in the static greedy algorithm. The proposed algorithm achieves the same accuracy with the state-of-the-art greedy algorithms while the number of Monte Carlo simulations needed is dramatically reduced by two orders of magnitude. We further give a dynamic update strategy to improve the static greedy algorithm, by applying which our algorithm becomes comparable

to the most scalable heuristic algorithm.

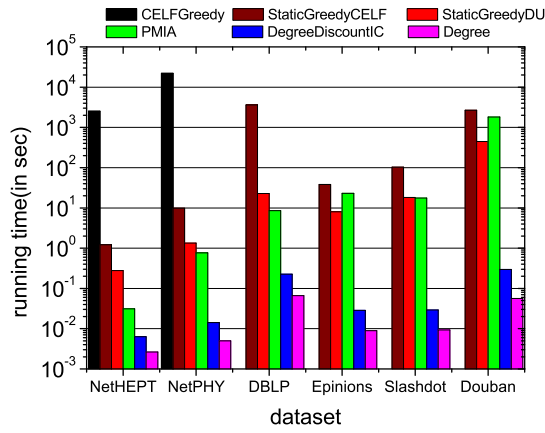
For the future work, we will study how to determine the minimum number R of Monte Carlo simulations for a specific network and a given spread model. We also want to investigate whether the proposed strategy can be extended to the linear threshold model. Finally, we will try to implement the proposed static algorithm towards the frame of parallel computing, which further improves the computational efficiency. We also look forward to see more applications of our algorithm on real world networks and practical scenarios.

7. ACKNOWLEDGMENTS

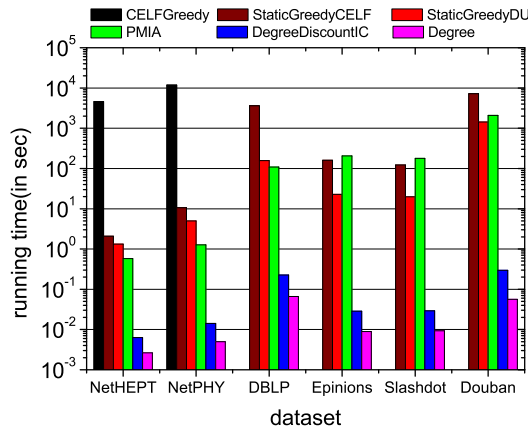
The authors thank Wei Chen for providing the codes of the PMIA algorithm and the datasets: NetHEPT, NetPHY and DBLP. The authors also thank to Jure Leskovec for providing the download for the social network datasets, Epinions and Slashdot. The authors thank to the member of the group NASC (www.groupnasc.org).

8. REFERENCES

- [1] P. Domingos, M. Richardson. Mining the network value of customers. In *Proceedings of the 7th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'01)*, pages 57–66, 2001.
- [2] M. Richardson, P. Domingos. Mining knowledge sharing sites for viral marketing. In *Proceedings of the 8th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'02)*, pages 61–70, 2002.
- [3] M. Kimura, K. Saito. Tractable models for information diffusion in social networks. In *Proceedings of the 10th European conference on Principle and Practice of Knowledge Discovery in Databases (PKDD'06)*, pages 259–271, 2006.



(a) UIC model



(b) WIC model

Figure 6: Running time comparison on test datasets under UIC model and WIC model.

- [4] J. Huang, X. Q. Cheng, J. Guo, H. W. Shen, K. Yang. Social recommendation with interpersonal influence. In *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI'10)*, pages 601–606, 2010.
- [5] J. Huang, X. Q. Cheng, H. W. Shen, T. Zhou, X. Jin. Exploring social influence via posterior effect of word-of-mouth recommendations. In *Proceedings of the 5th ACM International Conference on Web Search and Data Mining (WSDM'12)*, pages 573–582, 2012.
- [6] D. Kempe, J. M. Kleinberg, É. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the 9th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'03)*, pages 137–146, 2003.
- [7] E. Mossel and S. Roch. On the submodularity of influence in social networks. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 128–134, 2007.
- [8] W. Chen, C. Wang, Y. Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'10)*, pages 1029–1038, 2010.
- [9] W. Chen, Y. Yuan, L. Zhang. Scalable influence maximization in social networks under the linear threshold model. In *Proceedings of the 10th IEEE International Conference on Data Mining (ICDM'10)*, pages 88–97, 2010.
- [10] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, N. S. Glance. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'07)*, pages 420–429, 2007.
- [11] W. Chen, Y. Wang, S. Yang. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'09)*, pages 199–207, 2009.
- [12] M. Kimura, K. Saito, R. Nakano, H. Motoda. Extracting influential nodes on a social network for information diffusion. *Data Mining and Knowledge Discovery*, 20(1): 70–97, 2010.
- [13] Y. Wang, G. Cong, G. Song, K. Xie. Community based greedy algorithm for mining top-k influential nodes in mobile social networks. In *Proceedings of the 16th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'10)*, pages 1039–1048, 2010.
- [14] R. Narayanam, Y. Narahari. A shapley value-based approach to discover influential nodes in social networks. *IEEE Transactions on Automation Science and Engineering*, 8(1): 130–147, 2010.
- [15] Q. Jiang, G. Song, G. Cong, Y. Wang, W. Si, K. Xie. Simulated annealing based influence maximization in social networks. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI'11)*, pages 127–132, 2011.
- [16] A. Goyal, W. Lu, and L. V. S. Lakshmanan. CELF++: optimizing the greedy algorithm for influence maximization in social networks. In *Proceedings of the 20th International Conference on World Wide Web (WWW'11)*, pages 47–48, 2011.
- [17] Z. L. Luo, W. D. Cai, Y. J. Li, and D. Peng. A PageRank-based heuristic algorithm for influence maximization in the social network. *Recent Progress in Data Engineering and Internet Technology*, pages 485–490, 2012.
- [18] C. Wang, W. Chen, Y. Wang. Scalable influence maximization for independent cascade model in large-scale social networks. *Data Mining and Knowledge Discovery*, 25(3): 545–576, 2012.
- [19] M. Mathioudakis, F. Bonchi, C. Castillo, A. Gionis, A. Ukkonen. Sparsification of influence networks. *Proceedings of the 16th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'10)*, pages 529–537, 2011.
- [20] G. L. Nemhauser, L. A. Wolsey, M. L. Fisher. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming*, 8:73–87, 1978.